CMSCONSTRUCT

IT Discovery Machine

Content Gathering Module

UCMDB Integration

Chris Satterthwaite

Jan 22, 2019

Contents

Document Revisions

Author	Version	Date	Details
Chris Satterthwaite	1.0	1/14/2019	Document created
Chris Satterthwaite	1.1	1/21/2019	Added snapshots
Chris Satterthwaite	1.2	1/22/2019	Job names made more unique, in preparation for additional integrations for ITDM fingerprints

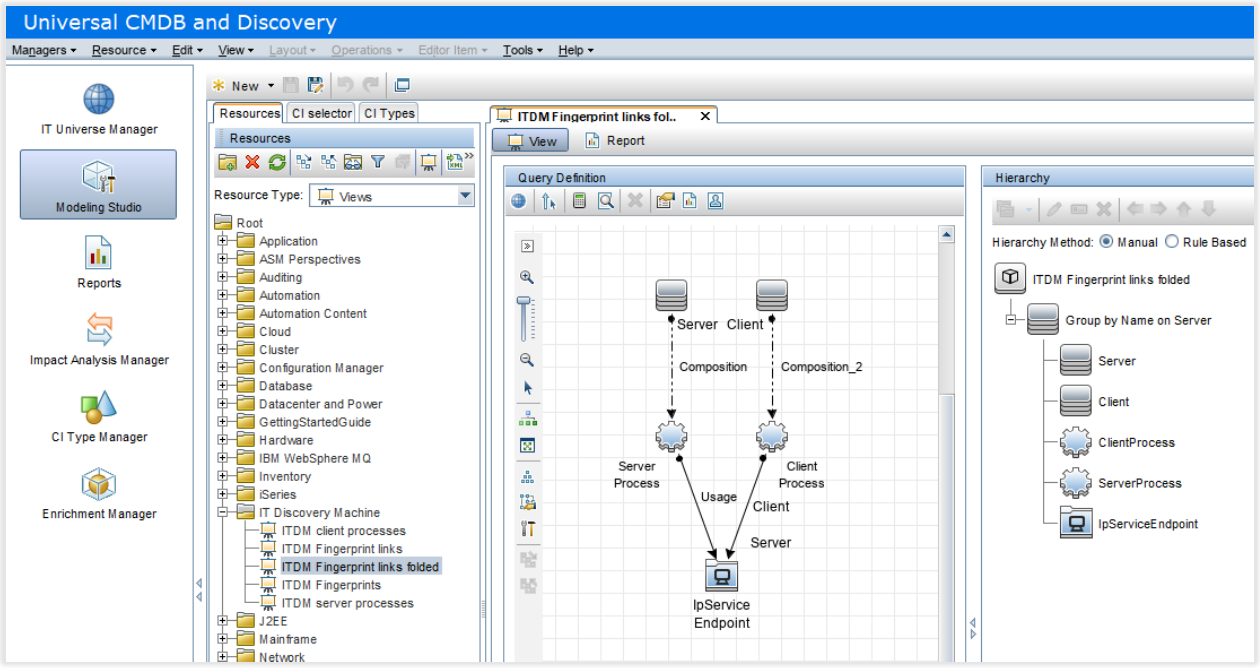
Introduction

This document details the 'integrationUCMDB' module written for IT Discovery Machine (ITDM). This module sends the following datasets into the UCMDB: Nodes discovered with ProcessFingerprints, mapped to related SoftwareFingerprints, and any TCP/IP dependencies between those objects.

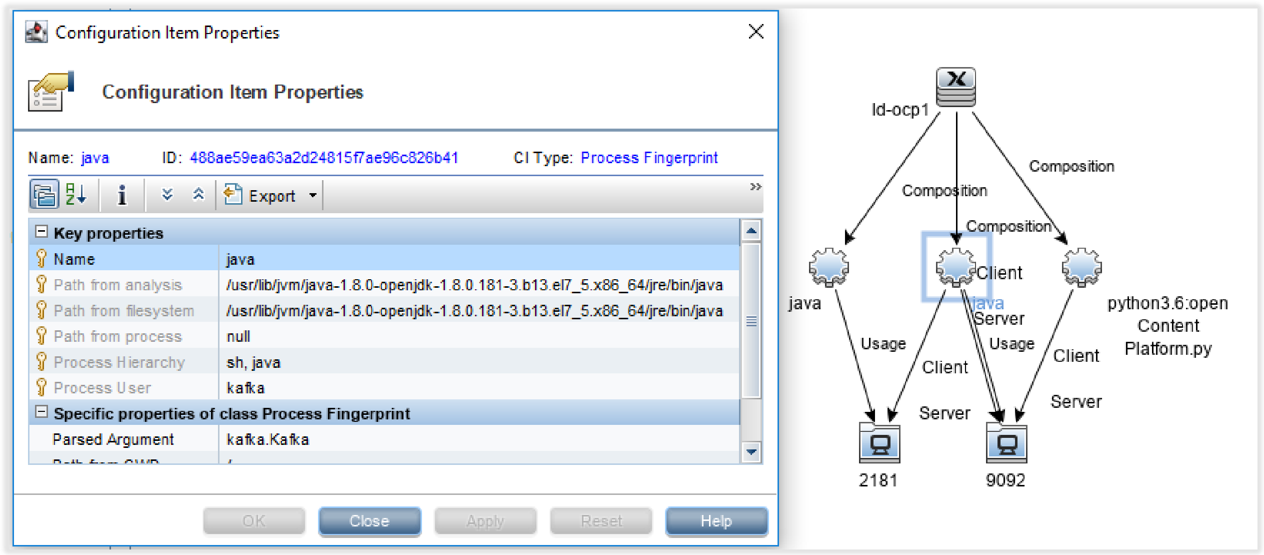
Sample Output

The snapshots in this section are accessible full size in the module's external/snapshots directory.

This first snapshot is from the UCMDB Modeling Studio, showing the five (5) packaged Views in the tree view on the left, along with one of them pulled up in the view pane on the right:



This snapshot shows internal process dependencies on a Linux server:

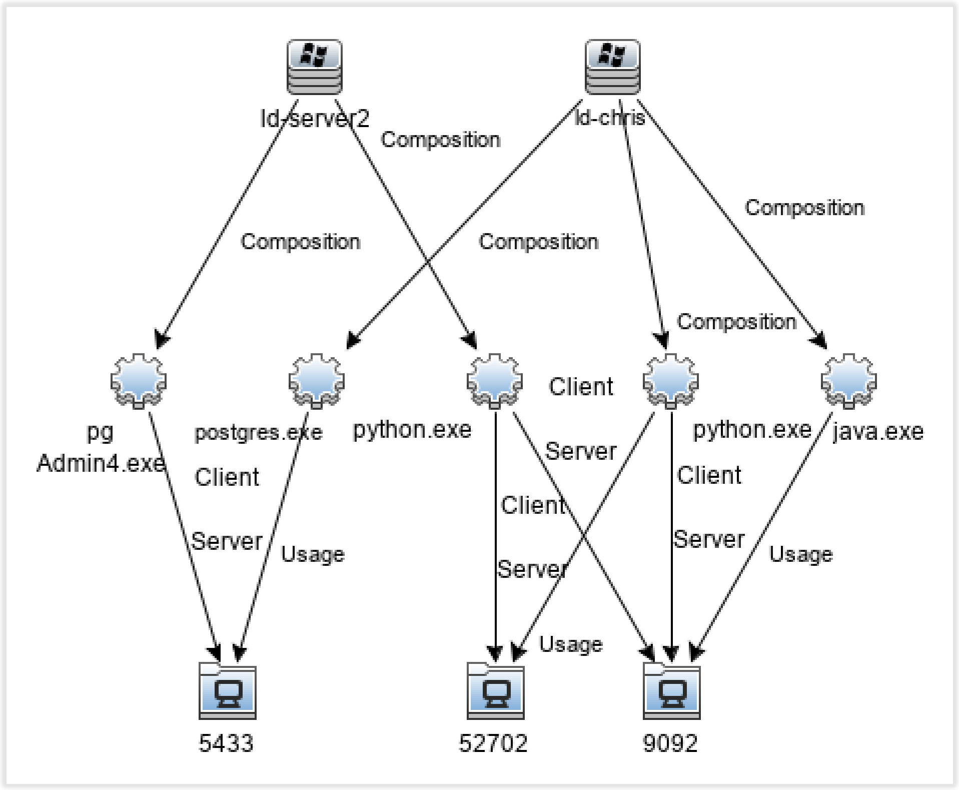


Configuration Item Properties

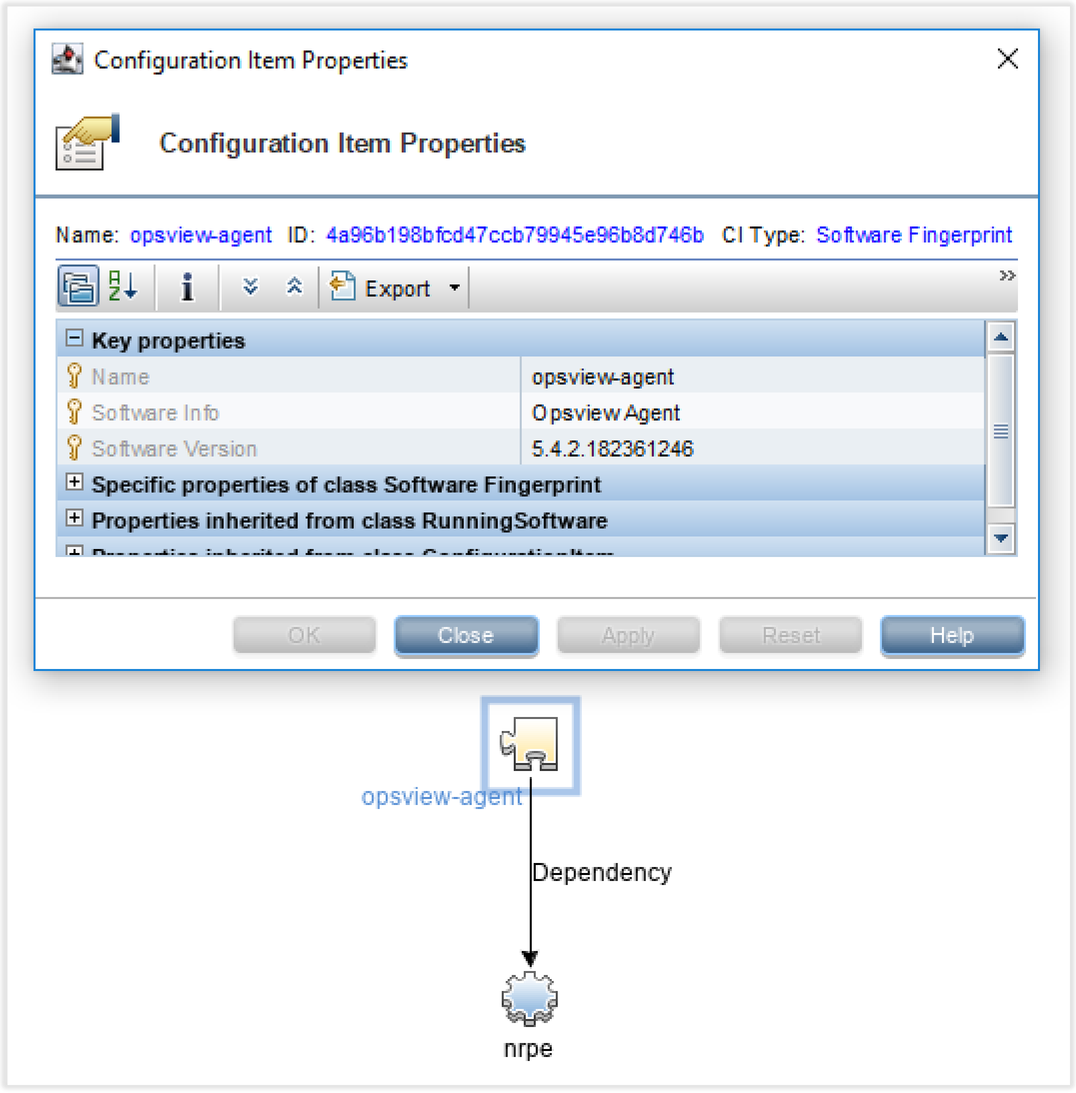
Name: java ID: 488ae59ea63a2d24815f7ae96c826b41 CI Type: Process Fingerprint

Key properties	
Name	java
Path from analysis	/usr/lib/jvm/java-1.8.0-openjdk-1.8.0.181-3.b13.el7_5.x86_64/jre/bin/java
Path from filesystem	/usr/lib/jvm/java-1.8.0-openjdk-1.8.0.181-3.b13.el7_5.x86_64/jre/bin/java
Path from process	null
Process Hierarchy	sh, java
Process User	kafka
Specific properties of class Process Fingerprint	
Parsed Argument	kafka.Kafka

The next one shows several dependencies between two (2) Windows endpoints:



The next snapshot illustrates an Opsview monitoring agent found on a Linux server. This captures the Properties of a Software Fingerprint CI, demonstrating some details brought in through the integration:



Configuration Item Properties

Configuration Item Properties

Name: opsview-agent ID: 4a96b198bfcd47ccb79945e96b8d746b CI Type: Software Fingerprint

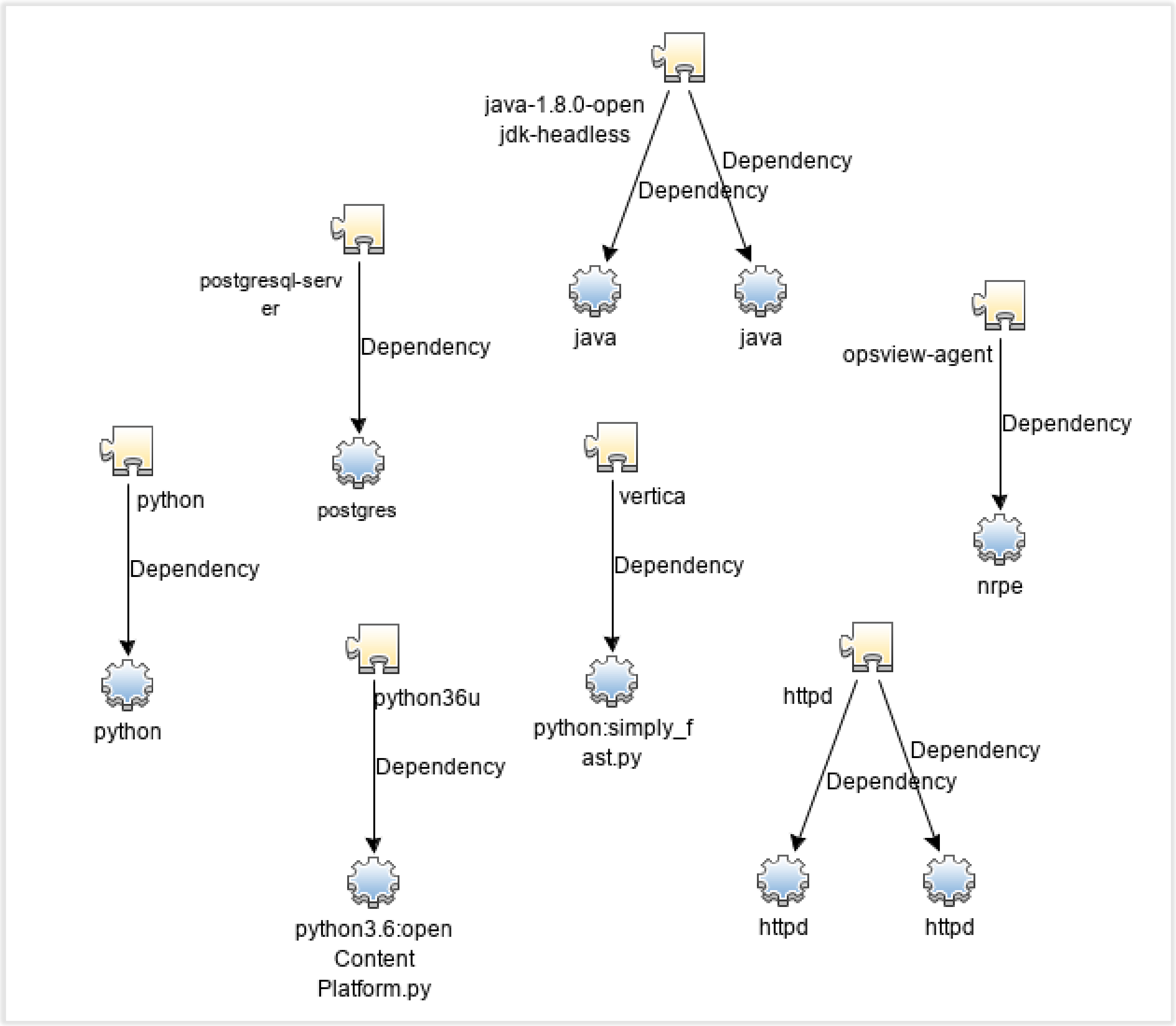
Key properties

Name	opsview-agent
Software Info	Opsview Agent
Software Version	5.4.2.182361246

Specific properties of class Software Fingerprint

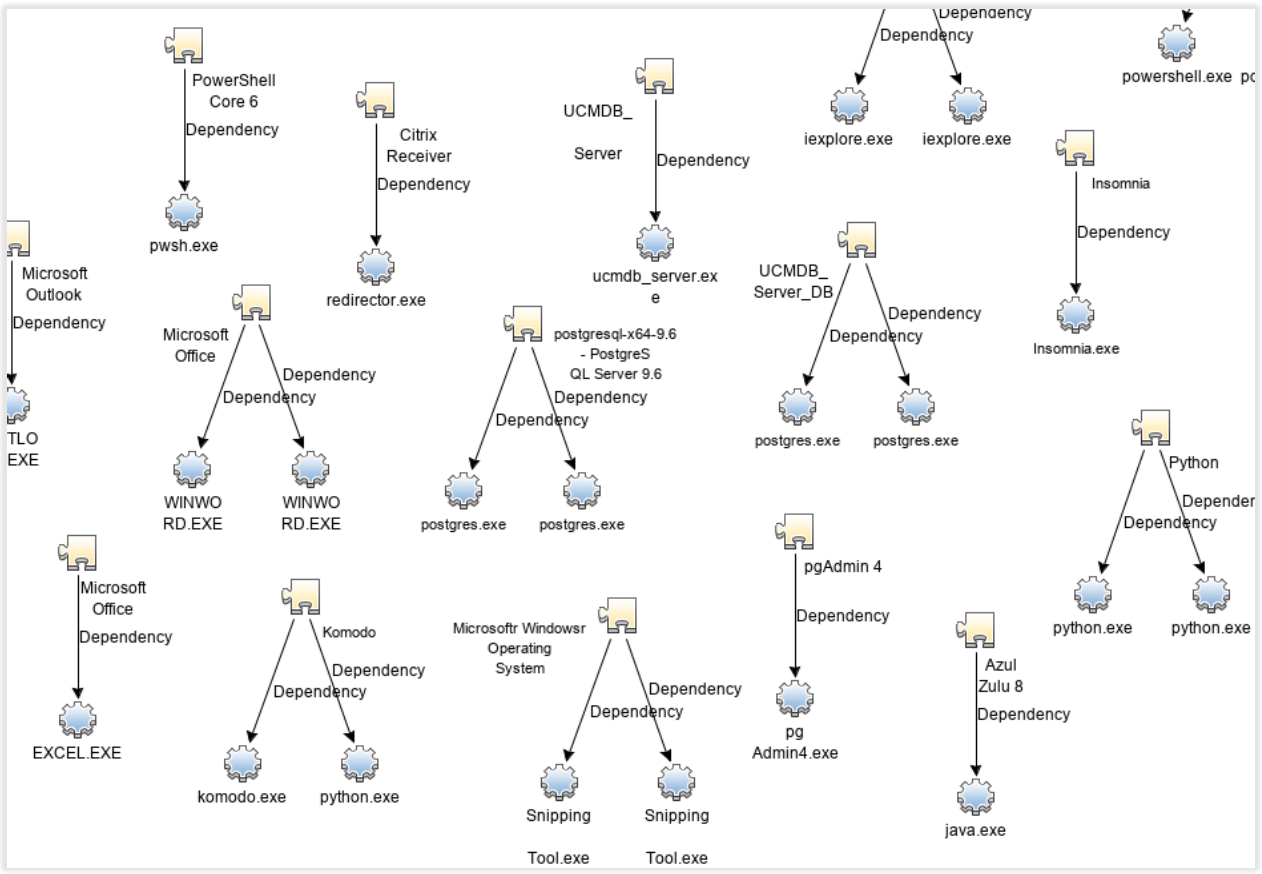
Properties inherited from class RunningSoftware

This shows the full list of Software components dynamically found on the previous Linux server:

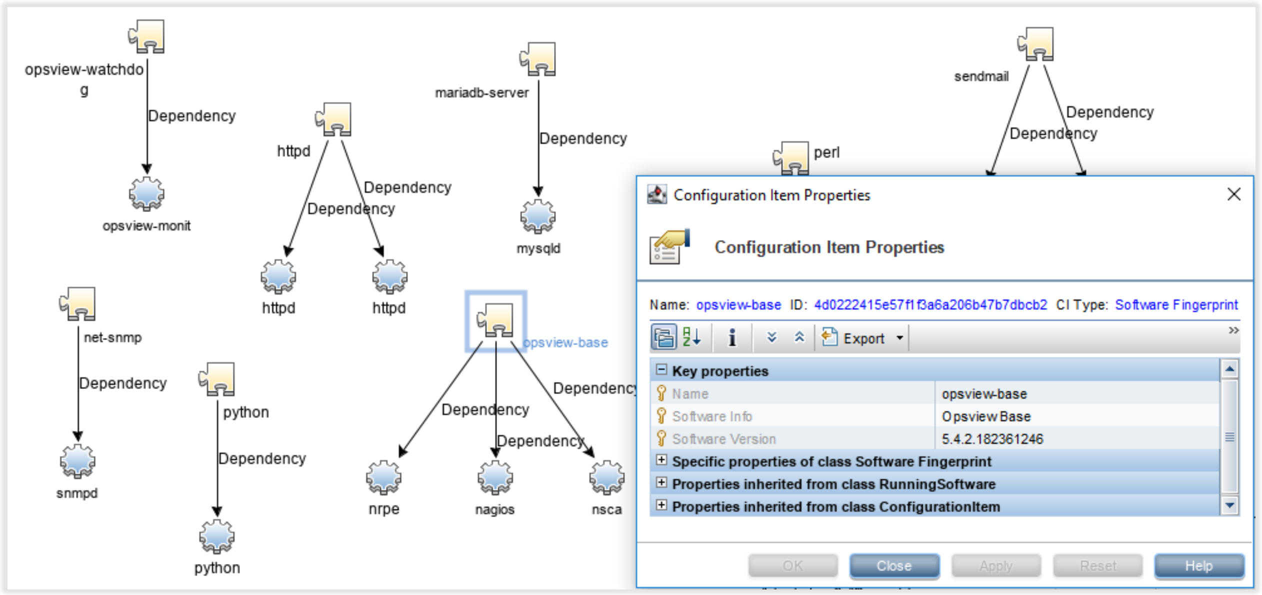


CMS CONSTRUCT

And similarly, this depicts Software components dynamically found on a Windows server:



The last snapshot shows software on another Linux server; one that is hosting Opsview Base:



Configuration Item Properties

Configuration Item Properties

Name: opsview-base ID: 4d0222415e57f1f3a6a206b47b7dbcb2 CI Type: Software Fingerprint

Export

Key properties	
Name	opsview-base
Software Info	Opsview Base
Software Version	5.4.2.182361246
Specific properties of class Software Fingerprint	
Properties inherited from class RunningSoftware	
Properties inherited from class ConfigurationItem	

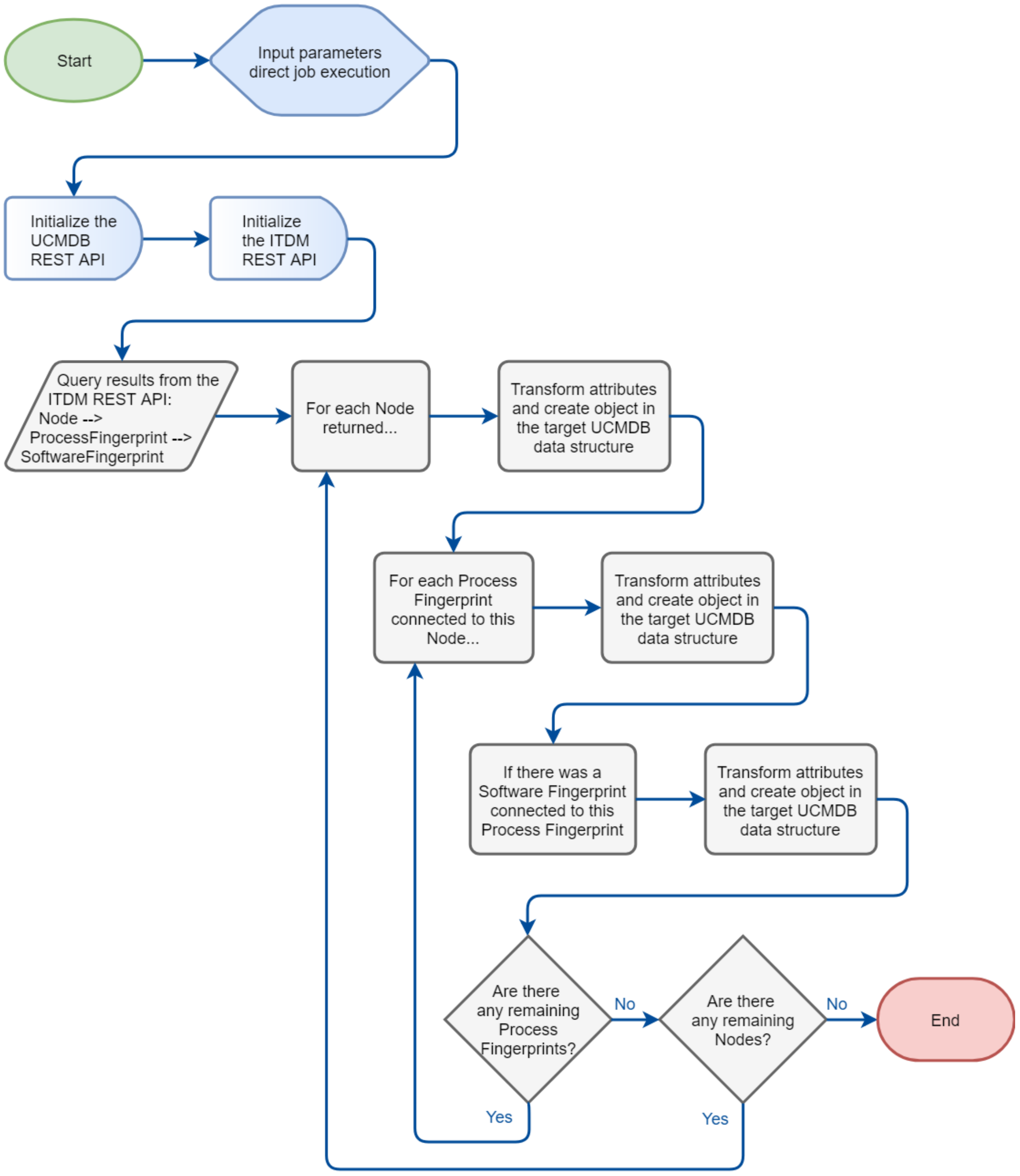
OK Close Apply Reset Help

CMS CONSTRUCT

Code Design

Flow Diagram

The following diagram shows the flow for the fingerprints job. The dependencies job processes two ITDM queries instead of one, which simply means it will have two result processing blocks:


Start
Input parameters direct job execution
Initialize the UCMDB REST API
Initialize the ITDM REST API
Query results from the ITDM REST API: Node --> ProcessFingerprint --> SoftwareFingerprint
For each Node returned...
Transform attributes and create object in the target UCMDB data structure
For each Process Fingerprint connected to this Node...
Transform attributes and create object in the target UCMDB data structure
If there was a Software Fingerprint connected to this Process Fingerprint
Transform attributes and create object in the target UCMDB data structure
Are there any remaining Process Fingerprints?
No
Yes
Are there any remaining Nodes?
No
Yes
End

CMS CONSTRUCT

Pseudocode

Pseudocode is an informal description of the design or algorithm of the code. This section takes the reader one level deeper into the working of the scripts. For specific functionality and code logic, please review the scripts themselves, which are written in Python with inline comments.

The 'itdm_to_ucmdb_fingerprints' job accomplishes the following:

1. Initialize the UCMDB REST API:
    a. Establish a REST connection via Python requests library.
    b. Provide username/password stored in an ITDM REST API credential for UCMDB access.
    c. Store the token to use in future communication, sending data to the UCMDB.
2. Initialize the ITDM REST API (seamlessly provided to content gathering jobs).
3. Query ITDM for Node-->ProcessFingerprints-->SoftwareFingerprint mappings:
    a. Name of query is pulled from the 'targetQuery' input parameter, and a file with this defined name must exist in the 'input' directory of this module. By default the targetQuery name is 'fingerprint_clients', which means the following file must exist:
       ./content/contentGathering/integrationUCMDB/input/fingerprint_clients.json
    b. Retrieve results via the 'getApiQueryResultsFull' utilities function, which is a blocking query instead of the related function 'getApiQueryResultsInChunks' that chunks the result set. Chunking is nice, but the chunking performed by the Query service currently (at the time of writing this job) constructs the result set in the 'Flat' format. And if we request the 'Nested-Simple' format instead, our scripts won't have to handle any post-processing (to group the results) before iterating through the data.
    c. Data set is returned in JSON format with a list of Nodes containing tree-layouts.
4. For each Node:
    a. Transform the Node attributes and create an object in the target UCMDB structure.
    b. Add the Node onto the result to send.
    c. For each Process Fingerprint related to this Node:
        i. Transform the Process Fingerprint into target UCMDB structure.
        ii. Add the Process Fingerprint onto the result to send
        iii. Add the Node to Process Fingerprint link
        iv. If there is a Software Fingerprint found for this Process Fingerprint:
            1. Transform the Software Fingerprint into target UCMDB structure
            2. Add the Software Fingerprint onto the result to send
    d. After the Node if fully processed, send results to UCMDB. This is done one Node at a time, in order to mitigate failures with insert/updates via data-in and reconciliation.
5. Update the ITDM job status and exit.

The 'itdm_to_ucmdb_fp_links' job accomplishes the following:

1. Initialize the UCMDB REST API:
    a. Establish a REST connection via Python requests library.

    b. Provide username/password stored in an ITDM REST API credential for UCMDB access.
    c. Store the token to use in future communication, sending data to the UCMDB.
2. Initialize the ITDM REST API (seamlessly provided to content gathering jobs).
3. Query ITDM **server** dependencies with ProcessFingerprints:
    a. Name of query is pulled from the 'serverQuery' input parameter.
    b. By default the value is 'fingerprint_servers', which means the following file must exist:
       ./content/contentGathering/integrationUCMDB/input/fingerprint_servers.json
    c. Retrieve results via the 'getApiQueryResultsFull' utilities function, for the same reason as described in the fingerprints job above.
    d. Data set is returned in JSON format with a list of Nodes containing tree-layouts.
4. For each Node:
    a. Transform the Node attributes and create an object in the target UCMDB structure.
    b. Add the Node onto the result to send.
    c. For each Process Fingerprint related to this Node:
        i. Transform the Process Fingerprint into target UCMDB structure.
        ii. Add the Process Fingerprint onto the result to send
        iii. Add the Node to Process Fingerprint link
        iv. For each TcpIpPort found for this Process Fingerprint:
            1. Transform the data into two related UCMDB objects:
                a. ip_address
                b. ip_service_endpoint
            2. Add the objects onto the result to send
            3. And relate to the Process Fingerprint
    d. After the Node if fully processed, send results to UCMDB.
5. Query ITDM **client** dependencies with ProcessFingerprints:
    a. Name of query is pulled from the 'clientQuery' input parameter.
    b. By default the value is 'fingerprint_clients', which means the following file must exist:
       ./content/contentGathering/integrationUCMDB/input/fingerprint_clients.json
    c. Retrieve results via the 'getApiQueryResultsFull' utilities function.
    d. Data set is returned in JSON format with a list of Nodes containing tree-layouts.
6. For each Node:
    a. Transform the Node attributes and create an object in the target UCMDB structure.
    b. Add the Node onto the result to send.
    c. For each Process Fingerprint related to this Node:
        i. Transform the Process Fingerprint into target UCMDB structure.
        ii. Add the Process Fingerprint onto the result to send
        iii. Add the Node to Process Fingerprint link
        iv. For each TcpIpPort found for this Process Fingerprint:
            1. Transform the data into two related UCMDB objects:
                a. ip_address
                b. ip_service_endpoint
            2. Add the objects onto the result to send
            3. And relate to the Process Fingerprint
7. After the Node if fully processed, send results to UCMDB.

8. Update the ITDM job status and exit.

Module Contents

Jobs

For general information on jobs (e.g. standard sections, parameter descriptions, general usage), refer to the Job Descriptor document.

There are two jobs available in this module:

File name	Description
itdm_to_ucmdb_fingerprints.json	Sends all ProcessFingerprint objects related to Node objects, along with any corresponding SoftwareFingerprint objects.
itdm_to_ucmdb_fp_links.json	Sends ProcessFingerprint objects related to Nodes, along with the related TCP/IP Port objects. This sends both client and server sides separately, which then overlay inside the UCMDB to provide the full picture.

Itdm_to_ucmdb_fingerprints

Main meta-data section:

```
"jobName" : "itdm_to_ucmdb_fingerprints",
"realm" : "default",
"clientGroup" : "default",
"protocolType" : "ProtocolRestApi",
"numberOfJobThreads" : 1,
"jobScript" : "send_fingerprints",
"clientOnlyTrigger" : true,
"clientEndpoint" : "any",
"endpointIdColumn" : "value",
```

Comments:

- This job is setup for a single destination, integration-type job. See the "Integration-type configuration" section of the Job Descriptor document for details.
- The protocolType directs the client to prepare all credentials for the ProtocolRestApi, since we will use a REST API credential in this job to connect into the UCMDB REST API.
- The jobScript is "send_fingerprints", meaning there is a file called "send_fingerprints.py" in the module's script directory.

Input parameters section:

```
"inputParameters" : {
		"ucmdbRestEndpoint" : "https://192.168.121.221:8443/rest-api",
		"credentialDescriptor": "UCMDB",
		"targetQuery" : "node_fingerprints",
		"printDebug" : false
	}
```

Parameter descriptions:

Parameter name	Description
ucmdbRestEndpoint	Base URL for the destination UCMDB REST API
credentialDescriptor	String to use in regEx matching on the protocol's description; this is how we direct the job to use the proper credential if there are many defined entries in the ProtocolRestApi table.
targetQuery	Set to node_fingerprints, meaning there is a file called "node_fingerprints.json" in the module's input directory.
printDebug	This enables logging specific to a job, instead of working off global log levels; when true, all runtime.logger.report() lines are printed.

Itdm_to_ucmdb_fp_links

Main meta-data section:

```
	"jobName" : "itdm_to_ucmdb_fp_links",
	"realm" : "default",
	"clientGroup" : "default",
	"protocolType" : "ProtocolRestApi",
	"numberOfJobThreads" : 1,
	"jobScript" : "send_fingerprint_links",
	"clientOnlyTrigger" : true,
	"clientEndpoint" : "any",
	"endpointIdColumn" : "value",
```

The only difference from the fingerprints job is with the values for jobName and jobScript.

Input parameters section:

```
	"inputParameters" : {
		"ucmdbRestEndpoint" : "https://192.168.121.221:8443/rest-api",
		"credentialDescriptor": "UCMDB",
		"serverQuery" : "fingerprint_servers",
		"clientQuery" : "fingerprint_clients",
		"printDebug" : false
	}
```

Since this job has two different queries to work through, it uses serverQuery and clientQuery, instead of a single targetQuery.

Input Queries

There are three (3) input queries available in this module:

File name	Description
node_fingerprints.json	Used in the itdm_to_ucmdb_fingerprints job; query for Node objects with related ProcessFingerprint objects, along with any corresponding SoftwareFingerprint objects.
fingerprint_clients.json	Used in the itdm_to_ucmdb_fp_links job; query for Node objects with **client** ProcessFingerprints related to TcpIpPorts.
fingerprint_servers.json	Used in the itdm_to_ucmdb_fp_links job; query for Node objects with **server** ProcessFingerprints related to TcpIpPorts.

Scripts

There are five (5) scripts available in this module. Two job entry points and three libraries:

File name	Description
send_fingerprints.py	Called by the itdm_to_ucmdb_fingerprints job; sends Node--ProcessFingerprints--SoftwareFingerprint mappings to the UCMDB
send_fingerprint_links.py	Called by the itdm_to_ucmdb_fp_links job; sends client/server ProcessFingerprints dependencies, from ITDM to the UCMDB.
fingerprintUtils.py	Library. Utilities shared between the two job entry scripts.
ucmdbTopologyData.py	Library. Contains the class for transforming objects/links into the UCMDB TopologyData structure.
ucmdbRestAPI.py	Library. Interface used in the job entry scripts to talk to the UCMDB REST API; think of this as a custom protocol wrapper.

External Files

There are two items in the external directory of this module. The first is the snapshots directory, which includes samples taken from the UCMDB. These were covered in "Sample Output" section above. The second entry in this folder is the UCMDB zip package: "itdmPkgForUCMDB.zip". It contains the following artifacts:

Folder name	Description
class	Contains two CIT class definitions: • process_fingerprint • software_fingerprint
identification	Contains two CIT overrides for the classes listed directly above, so they can use "primary keys" for reconciliation, instead of the standard "identification" rules.
cmdbview	Contains a folder (IT Discovery Machine) visible in the GUI, holding the five (5) different default views of the data sent from ITDM: • ITDM client processes • ITDM server processes • ITDM Fingerprints

	• ITDM Fingerprint links • ITDM Fingerprint links folded
tql	Contains the View folder, which contains query definitions required for the five (5) default Views listed directly above.
validlinks	Defines three (3) new links for this integration.

User Guide

Follow the steps in this section before enabling the job for the first time.

UCMDB REST API credentials

Work with the UCMDB team to gain username/password credentials you will use from ITDM to insert CIs and relationships into the UCMDB. After getting those, we need to provide them to ITDM:

The utility for this is found in the framework/lib sub-directory, and is called “createProtocolEntryUtil”.

Sample output follows from an execution of the utility:

```
PS D:\Software\CMS Construct\ITDM\server\framework\lib> python .\createProtocolEntryUtil.py

Protocol Type (sample entries: ['ProtocolSnmp', 'ProtocolWmi', 'ProtocolSsh', 'ProtocolPowerShell']):
ProtocolRestApi
Realm (valid entries: ['default', 'customDomain']): default
credential_group: default
active: 1
port:
description: UCMDB Dev environment
user: admin
password:
retype password:
```

Type something meaningful into the “description” part of the credential; that value is used along with the “credentialDescriptor” input parameter, to look up and match your desired credential.

This job doesn’t use the port; that’s provided by the value in the “ucmdbRestEndpoint” input parameter.

UCMDB package deployment for custom artifacts

This module requires a package be deployed on the UCMDB side, in order to create artifacts for this flow. The UCMDB-based package is called “itdmPkgForUCMDB.zip” and discussed in the External File section above. Deployment instructions follow:

1. Drop the Package in a directory accessible by the UCMDB Administrative GUI.
2. Deploy the package through the Package Manager:
    a. In the GUI, navigate to “Administration” and select “Package Manager”.
    b. Click the icon at the top, to “Deploy packages to server (from local disk)”.
    c. Click the plus button, then browse and select the package from step 1 above.
    d. Click “Deploy”, and wait for it to successfully deploy.

ITDM Job schedules

Open the two jobs defined in this module, and set the scheduler accordingly:

```
./content/contentGathering/integrationUCMDB/job/*.json
```

This type of setting will have a job run every Monday at 6:01 AM:

```
"triggerType" : "cron",
"triggerArgs" : {
        "year" : null,
        "month" : null,
        "day" : null,
        "week" : null,
        "day_of_week" : "mon",
        "hour" : 6,
        "minute" : 1,
        "second" : null,
        "start_date" : "",
        "end_date" : ""
},
```

Support Matrix

The following versions were tested with this module:

UCMDB version	Validated
10.33	✓
2018.11	✓

Troubleshooting

printDebug

The first step in troubleshooting is to enable the printDebug input parameter.

The printDebug parameter controls job-specific logging. When set to **true**, all runtime.logger.report() lines in job scripts are printed to the logs. Leaving this on all the time will consume unnecessary system resources, but enabling it is an obvious first step to troubleshooting:

```
"inputParameters" : {
        "printDebug" : true
}
```

UCMDB CI attributes with null or Unknown

The custom CI Types (Process Fingerprint and Software Fingerprint) have multiple fields marked as unique constraints in ITDM, meaning they are required to determine uniqueness. In ITDM, required constraint fields can be nullable, in which case the absence of a value is defaulted to an empty value (*null* in most languages, and *None* is Python).

When creating comparable classes in the UCMDB to hold those objects (custom CITs), those constraint fields are marked as "primary key". But unlike ITDM, the UCMDB will not allow those fields to be empty. If empty strings or a space character are used to fake it, reconciliation breaks and duplicates all instances of that CI.   If null is provided in the JSON (which is a valid setting in JSON to indicate empty value), the UCMDB REST API takes that value and converts it to a four character string: "null". While "null" strings are not so nice to see in the UCMDB attributes, they at least prevent mass CI duplication. So unfortunately you should expect to see "null" for empty fields in the UCMDB's representation of Process Fingerprints.

Now for the legacy "Unknown" value with Software Fingerprint versions. Depending on how it was populated, you may see "Unknown" in the version. This will go away after an upcoming enhancement, which will default Software Fingerprint versions to the empty value just like missing Process Fingerprint attributes. That means a string "null" will eventually replace the string "Unknown" values represented.